On the Comparison of Bacteriophage Populations

Anne Bergeron[1](✉), Marie-Jean Meurs[1], Romy Valiquette-Labonté[1], and Krister M. Swenson[2]

[1] Université du Québec à Montréal, Montreal, Canada
bergeron.anne@uqam.ca
[2] LIRMM, Université de Montpellier, CNRS, Montpellier, France

Abstract. The production of cheese and other dairy products relies on the constant monitoring of viruses, called bacteriophages, that attack the organisms responsible for the fermentation process. Bacteriophage species are characterized by a stable core genome, and a 'genetic reservoir' of gene variants that are exchanged through recombination. Phylogenetic analysis of phage populations are notably difficult due not only to extreme levels of horizontal exchange at the borders of functional modules, but also inside of them.

In this paper we present the first known attempt at directly modeling gene flux between phage populations. This represents an important departure from gene-based alignment and phylogenetic reconstruction, shifting focus to a genetic reservoir-based evolutionary inference. We present a combinatorial framework for the comparison of bacteriophage populations, and use it to compute recombination scenarios that generate one population from another. We apply our heuristic, based on this framework, to four populations sampled from Dutch dairy factories by Murphy [14]. We find that, far from being random, these scenarios are highly constrained. We use our method to test for factory-specific diversity, and find that there was likely a large amount of recombination in the ancestral population.

Find instructions for reproducing the results at:
https://bitbucket.org/thekswenson/phage_population_comparison
The code is publicly available at:
https://bitbucket.org/thekswenson/phagerecombination

1 Introduction

Bacteriophages – or simply *phages* – are viruses that infect bacteria. They are the most abundant and diverse organisms on the planet, and are found in every community where bacteria thrive: soil, water, air, lungs, guts, sewers, plants, and milk [9]. Where their presence intersects human activity, they can be beneficial, when they are used in therapies to combat bacterial infections [4], or detrimental, when they destroy batches of dairy fermentation in artisanal or industrial food factories [13]. Due to their economic impacts, dairy bacteriophage populations

L. Jin and D. Durand (Eds.): RECOMB-CG 2022, LNBI 13234, pp. 3–20, 2022.
https://doi.org/10.1007/978-3-031-06220-9_1

have been extensively sequenced in the last few years. These populations can be separated by geography [5,6,14], by time [10,11], or by their bacterial host [3].

Bacteriophages are divided into *species*, characterized by a common *core* genome distributed along their single chromosome, where genes appear in the same order for each member of the species. Between these regions of core genome, there is a *variable* genome composed of regions that are shared by some members of the family, but not all of them. In an individual phage, the variable region between two consecutive regions of core genome may be empty, or may have one or more *variants* that are presumed to perform the same biological function, but with different proteins [3, & references therein].

Kupczok [10] sequenced 34 dairy phages from a single German dairy factory, sampled over three decades. Their analyses concluded that, over such a period of time, point mutations were "*[...] unlikely to constitute the major driver of phage genome evolution*". However, the variable genome of the sequenced phages changed considerably over time: "*The frequent gene loss and regain suggest the existence of a pangenome (i.e. genetic reservoir) that is accessible by genetic recombination.*"

1.1 Recombinations and Mosaicism in Phage Genomes

Genetic *recombination* allows two phages to exchange or borrow significant parts of their genomes, creating novel viruses. These exchanges take place inside a single cell, and are presumed to occur either between two co-infecting phages, or by an infecting phage and a *prophage* (*i.e.* a phage genome that inserted itself into a host bacterium). When the exchanges occur between similar sequences, recombinations are called *homologous*, and when they occur between unrelated sequence they are called *illegitimate*.

A striking feature of the comparison of phage genomes is their extreme *mosaicism*, where regions of unrelated sequences alternate with regions of very high similarity, as illustrated in Fig. 1.

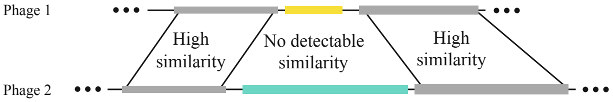


Fig. 1. Alignments of phage genomes exhibit alternating regions of high local similarity and unrelated regions.

A few decades ago, Botstein [2] proposed a *theory of modular evolution* for bacteriophages based on homologous recombinations. In this model, recombinations are mediated by flanking regions of high sequence similarity. DePaepe [7] characterized biological mechanisms that could be responsible for such rearrangements, calling them *relaxed homologous recombinations*, and qualifying them as *"...strangely dependent on the presence of sequence homology, but highly tolerant to divergence"*.

Recombination that does not follow the Botstein model also likely plays an important role in phage evolution. Pedulla [15] found three recent recombinations in Mycobacteriophages that have no flanking regions of similarity. Yahara [18] analyzed recombination within *soft-core* genes (*i.e.* the genes that exist in at least 90% of the phages they studied) of *Helicobacter pylori* prophages, showing extreme sequence divergence that they attribute to recombination within genes.

Thus, phages are particularly ill suited to traditional evolutionary analyses using multiple sequence alignment. The extraordinary mosaicism implies that not any set of genes with the same function can be used for phylogenetic inference, since in this case shared function does not imply homology. Instead, Brussow [3] recommends that evolutionary histories should be established using only the homologous sequences belonging to the same functional module (*i.e.* what we call a *variant* in the present article). Yet, to make matters worse, the high dissimilarity attributed to pervasive recombination within genes makes phylogenetic reconstruction on individual variants very difficult [18].

1.2 Recombination Between Phage Populations

The conditions of high mosaicism in bacteriophages motivate a fresh perspective on evolutionary history inference, one that uses the fact that a population of phages represent a genetic reservoir that is constantly testing combinations. To this end, in a previous work we inferred a recombination scenario within a population of phages while explicitly using the Botstein model of recombination with flanking homologous regions [17]. The present article differs from our previous work in two important ways:

1. here we infer recombination scenarios *between* phage populations instead of *within* them, and
2. our present model can accomodate flanking homology or ignore it.

When building our modules we assume the existence of flanking homology "anchors" between every adjacent module (see Sect. 3.1 for details on how we constructed our modules).

Our work is timely, given that *"little is known about genetic flux by recombination between populations"* [18]. While it represents a first attempt at reconstructing the evolution of phages in such a global sense, we expect this perspective to increase in importance as phage sequencing becomes more prevalent.

Dutch Dairy Factories. In Murphy [14], phages are sampled across geographic regions. They sequenced 38 phage genomes from four Dutch dairy factories, and added to their dataset phages of the same species from various countries and continents (Australia, Canada, Denmark, France, Germany, Ireland, Italy, Poland, United Kingdom, United States, and New Zealand). By using hierarchical clustering on protein families presence/absence data, they were able to – mostly – separate continents and countries. However, this technique was not able to separate the four Dutch factories.

```
Factory 1
A: A---AAAA-A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAAAAA--AAA-AAAA-A-AA---A--A-A--A--AAAA-A
B: B---BAA-BABA-AABA-A-B-B-BBB-BABBABAB-BB------BBABBBBBA-A--B-B-BB-----BB-BBBB-BBAAA-A
C: B---BA-A-A-A-AACA-CC--C--CA-BCCCACCCCBC-CCCC-CCACCCBBACA--CCCCCC-C---CACC--CC-C-CACA
D: D---BA-D-A-D-AABA-ACD-D-BCA-DADDDDADDD-------DDABD-DBA-A-DD-ADDD-----BB-D--DC-DADADA
E: B-E-BAA--A-AEAACE-AC--E--C--EAEEECAE-B------EEEAECCBBA-E--E-A-EE-C--E-A-E--E--EAEADE
F: B---BA-A-A-A-AACA-CCF-C--CA-BCCCACCCCBC-CCCC-CCAFCCBBACA--CCCCFC-C---CACC--CC-F-CACA
G: D---BA-D-A-D-AABA-AC--D-BCA-DADDEDAGDD-------DDABD-DBA-A-DD-ADDD-----BB-G--D--GADADA
H: BH--BA-A-A-A-AACA-CC--C--CA-BCCCACCCCBC-CCCC-CCAHCCBBACA--CCCCHH-CH--BB-H--CC-HAEACA
I: B-E-BAA--A-AEAACE-AC--E--C--EAEIECAE-B------EEIAECCBBA-E--I-A-EE-C----A-I--E--I-EA-E
J: A---AAAA-A-A-AAAA-AJ---A-AA-AAAAAAAAAA----AAAJAAA--AAA-AAAJ-A-JJ---A--A-A--J--JAAA-A
K: D---BA-D-A-D-AABA-AC--D-BCA-DADDDDADDD-------DDABD-DBA-A--K-ADKK---KKBB-K--DC-KAAADA
L: L---BAL----A-A-BAL-C-L---LA-DALLDLAL-B------ELLALL-DBA-L-DL-A-L--L----A-L--L--LAAADA
M: A---AAAA-A-A-AAAA-AJ--MA-CA-AAAAAAAAAA----AAAMAAM-MMBM-AAAJ-A-MA---A--A-MM-J--MAAA-A
Factory 2
N: N---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAMAAN-MMBM-AAAJ-A-MJ---A--A-N--A--NAAA-A
O: O---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAMAAO-MMBM-AAAJ-A-OO---A--A-O--A--OAAA-A
P: P---BAAA-A-A-AAAE-AC--E--C--EAEPECAE-B------EEEAECCBBA-E--E-A--P-CH---A-P--E-BP-EA-E
Q: BH--BA-A-A-A-AAC--CC--C--CA-BCCQACCCCBC-CCCC-CCAQCCBBACA--CCCCQH-CH--BB-Q--CC-QAEACA
Factory 3
R: A---AAAA-A-A-AAAA-AJ--RA-CA-AAAAARAAAA----AAARAAA--AAA-AAAR-A-AJ---A--A-R--A--RAAA-A
S: A---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAASAAA--AAA-AAAJ-A-AA---A--A-A--A--SAAA-A
T: L---BAA-BABA-AABA-A-B-T-BBB-BABBABAB-BB------TTATL-DBA-L--T-B-TT----TBB-T--B--TAAA-A
U: L---BAA-BABA-AABA-A-B-T-BBB-BABBABAB-BB------TUAUBBBBA-L--T-B-TT----TBB-T--B--UAAA-A
V: L---BAA-BABA-AABA-A-B-T-BBB-BABBABAB-BB------TTAVBBBBA-L--T-B-TT-----BB-VBBB--VAAA-A
W: W---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAWAAA--AAA-AAAJ-A-AJ---A--A-A--J--WAAA-A
X: X---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAMAAX-MMBM-AAAJ-A-OO---A--A-X--A--XAAA-A
Y: Y---B--D-A-A-AABALA-Y-----A-AAYYAYA---C------YYAAD-YB--A-AY-A-Y------BB-Y--Y--YAA--A
Z: B---BAA-BABA-AABA-A-B-Z-BBB-BABBABAB-BB------BBABBBBBA-A--B-B-ZB-----BB-BBBB-ZZAAA-A
a: B---BAA-BABA-AABA-A-B-a-BBB-BABBABAB-BB------BBABBBBBA-A--B-B-aB-----BB-BBBB-ZaAAA-A
b: b---AAA--AbA-AABALA------bA-AAAbAbAbAA----AAAbbAA--AAA-AAAb-A-bb------A-b--A--bAAA-A
c: L-E-BAL-cAbA-AAcALA-B---Bc--BCcc---c-BB-----ELcAcBBBBA-A--c-A--c-C---BB-c--CCBcAAA-A
d: A---AAAA-A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAMAAA--AAA-AAAJ-A-ddd--A--A-d--J--dAAA-A
Factory 4
e: D---BA-D-A-D-AABA-AC--D-BCA-DADDDDAeDD-------DDABD-DBA-A--K-ADee---KK-A-e--D--eAAA-A
f: L---BA-A-A-A-AACA-CC--C--CA-BCCCACCCCBC-CCCC-CCAfCCBBACA--CCCCCf-CH--BBCf--CC-f-CACA
g: B--gBA-gcA-A-gABA-A--Lg--gAg-AgggCAggB-g----EggAgLgDBACL--g-A-gg-C---BB-g--g--g-AADA
h: A---AAAA-A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAMAAA--AAA-AAAJ-A-ddd--A--A-d--J--hAAA-A
i: B--gBA-gcA-A-gABA-A--Lg--gAg-AgggCAggB-g----EgiAiLgDBACL--g-A-gg-C---BB-g-Bg--iAAADA
j: j---B--A-A-A-A-BALA---j---A--AjjACAjDD-------D--jD-DBA-A--j-A-jj-C-A-BB-jj-CC-jAAA-A
k: L---AAAA-A-A-AAAA-AJ---A-AA-AAAAAAAAAA----AAAMAAA--AAA-AAAk-A-kbd--A--A-k--J--kAAA--
l: O---AA---A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAllAA--AAA-AAAk-A-lbd--A--A-l--J--l-AA-A
```

(a) Grouping by factories

```
Group A
A: A---AAAA-A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAAAAA--AAA-AAAA-A-AA---A--A-A--A--AAAA-A
J: A---AAAA-A-A-AAAA-AJ---A-AA-AAAAAAAAAA----AAAJAAA--AAA-AAAJ-A-JJ---A--A-A--J--JAAA-A
M: A---AAAA-A-A-AAAA-AJ--MA-CA-AAAAAAAAAA----AAAMAAM-MMBM-AAAJ-A-MA---A--A-MM-J--MAAA-A
N: N---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAMAAN-MMBM-AAAJ-A-MJ---A--A-N--A--NAAA-A
O: O---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAMAAO-MMBM-AAAJ-A-OO---A--A-O--A--OAAA-A
R: A---AAAA-A-A-AAAA-AJ--RA-CA-AAAAARAAAA----AAARAAA--AAA-AAAR-A-AJ---A--A-R--A--RAAA-A
S: A---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAASAAA--AAA-AAAJ-A-AA---A--A-A--A--SAAA-A
W: W---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAWAAA--AAA-AAAJ-A-AJ---A--A-A--J--WAAA-A
X: X---AAAA-A-A-AAAA-AJF--A-AA-AAAAAAAAAA----AAAMAAX-MMBM-AAAJ-A-OO---A--A-X--A--XAAA-A
d: A---AAAA-A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAMAAA--AAA-AAAJ-A-ddd--A--A-d--J--dAAA-A
h: A---AAAA-A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAMAAA--AAA-AAAJ-A-ddd--A--A-d--J--hAAA-A
k: L---AAAA-A-A-AAAA-AJ---A-AA-AAAAAAAAAA----AAAMAAA--AAA-AAAk-A-kbd--A--A-k--J--kAAA--
l: O---AA---A-A-AAAA-A----A-AA-AAAAAAAAAA----AAAllAA--AAA-AAAk-A-lbd--A--A-l--J--l-AA-A
Group B
B: B---BAA-BABA-AABA-A-B-B-BBB-BABBABAB-BB------BBABBBBBA-A--B-B-BB-----BB-BBBB-BBAAA-A
T: L---BAA-BABA-AABA-A-B-T-BBB-BABBABAB-BB------TTATL-DBA-L--T-B-TT----TBB-T--B--TAAA-A
U: L---BAA-BABA-AABA-A-B-T-BBB-BABBABAB-BB------TUAUBBBBA-L--T-B-TT----TBB-T--B--UAAA-A
V: L---BAA-BABA-AABA-A-B-T-BBB-BABBABAB-BB------TTAVBBBBA-L--T-B-TT-----BB-VBBB--VAAA-A
Z: B---BAA-BABA-AABA-A-B-Z-BBB-BABBABAB-BB------BBABBBBBA-A--B-B-ZB-----BB-BBBB-ZZAAA-A
a: B---BAA-BABA-AABA-A-B-a-BBB-BABBABAB-BB------BBABBBBBA-A--B-B-aB-----BB-BBBB-ZaAAA-A
Group C
C: B---BA-A-A-A-AACA-CC--C--CA-BCCCACCCCBC-CCCC-CCACCCBBACA--CCCCCC-C---CACC--CC-C-CACA
F: B---BA-A-A-A-AACA-CCF-C--CA-BCCCACCCCBC-CCCC-CCAFCCBBACA--CCCCFC-C---CACC--CC-F-CACA
H: BH--BA-A-A-A-AACA-CC--C--CA-BCCCACCCCBC-CCCC-CCAHCCBBACA--CCCCHH-CH--BB-H--CC-HAEACA
Q: BH--BA-A-A-A-AAC--CC--C--CA-BCCQACCCCBC-CCCC-CCAQCCBBACA--CCCCQH-CH--BB-Q--CC-QAEACA
f: L---BA-A-A-A-AACA-CC--C--CA-BCCCACCCCBC-CCCC-CCAfCCBBACA--CCCCCf-CH--BBCf--CC-f-CACA
Group D
D: D---BA-D-A-D-AABA-ACD-D-BCA-DADDDDADDD-------DDABD-DBA-A-DD-ADDD-----BB-D--DC-DADADA
G: D---BA-D-A-D-AABA-AC--D-BCA-DADDEDAGDD-------DDABD-DBA-A-DD-ADDD-----BB-G--D--GADADA
K: D---BA-D-A-D-AABA-AC--D-BCA-DADDDDADDD-------DDABD-DBA-A--K-ADKK---KKBB-K--DC-KAAADA
e: D---BA-D-A-D-AABA-AC--D-BCA-DADDDDAeDD-------DDABD-DBA-A--K-ADee---KK-A-e--D--eAAA-A
Group E
E: B-E-BAA--A-AEAACE-AC--E--C--EAEEECAE-B------EEEAECCBBA-E--E-A-EE-C--E-A-E--E--EAEADE
I: B-E-BAA--A-AEAACE-AC--E--C--EAEIECAE-B------EEIAECCBBA-E--I-A-EE-C----A-I--E--I-EA-E
P: P---BAAA-A-A-AAAE-AC--E--C--EAEPECAE-B------EEEAECCBBA-E--E-A--P-CH---A-P--E-BP-EA-E
```

(b) Grouping by genome organization

Fig. 2. (a) The variable parts of the 38 phage genomes of Murphy [14], color-coded by factory. Each letter stands for a variant spanning the interval between two regions of the core genome. (b) Highly similar genome organization is observed in 31 of the 38 phage genomes. Using the color-coding of panel (a), we see that each of the 5 groups has a representative in at least two different factories.

Figure 2a shows the variable parts for the 38 phage genomes of [14], color-coded by factory. Each letter stands for a variant spanning the interval between two consecutive *anchors* (*i.e.* regions of core genome), and dashes represent empty variants. The phages were grouped by similar genome organizations and variants. Seven of these phages were in groups occurring in only a single factory, having few shared modules with the other factories, so were removed from consideration. Figure 2b shows the 31 remaining phages. The comparison of Figs. 2a and 2b implies that each factory hosts a crew of different phages, that is more or less conserved across factories, suggesting a common ancestral population.

We apply the algorithm described in this paper to the populations from each pair of the Dutch factories. The lengths of the calculated recombination scenarios are used to infer relative properties of population diversity between the factories. Experiments are performed to determine if factories have specific qualities, and to demonstrate a likely high amount of ancestral recombination within phage populations.

Paper Outline. In this paper, we introduce the concept of recombination scenarios, describing how a phage population can be derived from another.

The next section gives the basic definitions and properties, Sect. 2.2 presents the theoretical basis of the greedy heuristics, and Sect. 2.3 derive lower bounds adapted to specific characteristics of biological data. Finally in Sect. 3, our heuristic is used to compare the four dairy factories of Fig. 2a.

2 Methods

2.1 Basic Definitions and Properties

A *phage species* is a set of phage genomes whose core genome contains the same number m of distinct regions, called *anchors*, thus the same number m of variable regions called *modules*. Each module has two or more *variants* within the species. In this paper, we work with circularized versions of phage genomes[1].

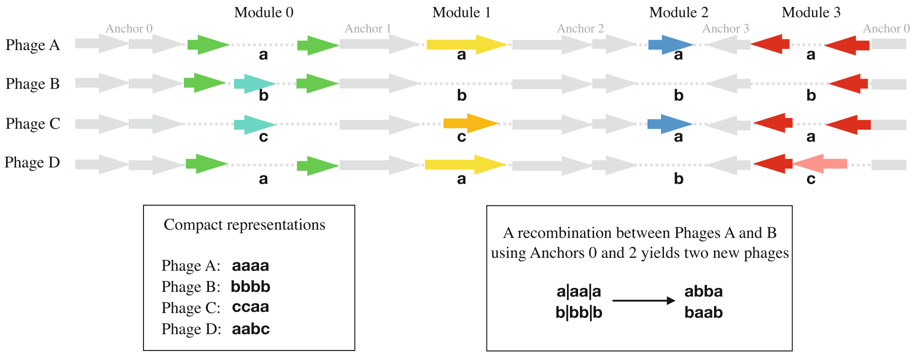


Fig. 3. Each phage in a set can be represented by the sequence of its variants. For each module, an arbitrary symbol is assigned to each variant, including empty variants. This compact representation captures the different assortments of modules within the population. Intervals are sequences of consecutive modules in the circular order. A recombination exchanges intervals between two parents, creating two new phages.

By representing variants of a module by single symbols such as a, b, c, …, it is possible to represent individual members of a species by the sequence of their variants, as in Fig. 3. Anchors are numbered from 0 to $m - 1$ in the clockwise direction, where $m \geq 2$ is the number of modules.

More formally, given sets of variants $\mathcal{V}_i$ for each module i, a phage p can be represented by $p = p_0 p_1 \dots p_{m-1}$ where $p_i \in \mathcal{V}_i$. The *recombination* operation at anchors a and b between two phages p and q yields new phages c and d:

$$\begin{aligned} p &= p_0 p_1 \dots p_{a-1} | p_a \dots p_{b-1} | p_b \dots p_{m-1} \\ q &= \; q_0 q_1 \dots q_{a-1} | q_a \dots q_{b-1} | q_b \dots q_{m-1} \end{aligned}$$

yields

[1] After invading a cell, linear phage genomes are often circularized. This is due to a variety of mechanisms: a circular configuration may protect phages from degradation by the defense mechanisms of the bacteria; it may allow the phage genome to be duplicated as a plasmid, or to be integrated in the host genome; or it may be used to initiate a *rolling circle* replication procedure that leads to a concatenamer [16].

$$
\begin{aligned}
c &= p_0 p_1 \dots p_{a-1} | q_a \dots q_{b-1} | p_b \dots p_{m-1} \\
d &= \; q_0 q_1 \dots q_{b-1} | p_a \dots p_{b-1} | q_b \dots q_{m-1}.
\end{aligned}
$$

The recombining phages are called *parents*, and the newly constructed phages, their *children* or *descendants*, the pair $\{c, d\}$ is a pair of *twins*. Comparisons between phage populations are based on the following relation, which is the main focus of this paper:

Definition 1. *Given two populations P and Q, a* recombination scenario *from P to Q is a sequence of recombinations that constructs all phages of Q using phages of P and their descendants. When there exists at least one recombination scenario from P to Q, we say that P* generates *Q, and we write $P \rightarrow Q$. The number of recombinations in a shortest scenario is ℓ_{PQ}.*

There is a simple way to check whether $P \rightarrow Q$. Indeed we have:

Proposition 1. *The relation $P \rightarrow Q$ holds if and only if, for each module, every variant that appears in Q also appears in P.*

If follows from Proposition 1 that the existence of $P \rightarrow Q$ does not imply the existence of $Q \rightarrow P$, since certain variants of modules of population P may have been lost in the recombination process. The operation $P \rightarrow Q$ has the following properties:

Proposition 2. *For any population P, $P \rightarrow P$ and $\ell_{PP} = 0$. If $P \rightarrow Q$ and $Q \rightarrow R$, then $P \rightarrow R$, and $\ell_{PR} \leq \ell_{PQ} + \ell_{QR}$.*

Since the measure ℓ_{PQ} is not symmetric, it is not a distance, but it is always possible to convert it to a distance by considering the sum $\ell_{PQ} + \ell_{QP}$. However, with actual biological data, it turns out to be much more interesting to compare ℓ_{PQ} and ℓ_{QP}.

The central problem that we address in this paper is the following:

Problem 1. Given phage populations P and Q such that $P \rightarrow Q$, compute a recombination scenario of length ℓ_{PQ}.

The computational complexity of Problem 1 is currently unknown, even when $|Q| = 1$. The main theoretical hurdle is that the notion of *breakpoint*, which is central to most genome rearrangement problems, is not well-defined: it is often impossible to determine *a priori* the number, nature, or positions of breakpoints.

In order to develop approximate solutions, we need objective functions that are guaranteed to decrease at each iteration of the process, these are developed in Sect. 2.2, where a first greedy heuristics is outlined. Evaluating the performance of the heuristics requires theoretical lower bounds for the length of a recombination scenario. These bounds are derived in Sect. 2.3.

2.2 Minimum Covers

We define a *circular interval* $(s..t)$ as the subset of integers $\{s, s+1, \ldots, t\}$, where additions are done modulo m. An *interval* in a phage p is denoted by $p(s..t)$; a phage interval $p(s..t)$ is *contained* in a phage interval $q(s'..t')$ if the circular interval $(s..t)$ is contained in $(s'..t')$, and $p_k = q_k$ for all k in $(s..t)$.

In particular, the equality of phage intervals $p(s..t) = q(s..t)$ implies that they share the same modules with the same variants.

Definition 2 (Covers and minimum covers)
Let P be the population of parents, and Q the population of children. A cover *of a child c is a set $\mathcal{C}(c) = \{p^1(s_1..t_1), p^2(s_2..t_2), \ldots, p^n(s_n..t_n)\}$ of n intervals, where each phage $p^k \in P$, and such that:*

1. *The union $\bigcup_{k\in\{1..n\}} p^k(s_k..t_k)$ is equal to c;*
2. *No interval in $\mathcal{C}(c)$ is contained in another interval of $\mathcal{C}(c)$.*
3. *Each interval $p^k(s_k..t_k)$ is* maximal, *in the sense that neither $p^k(s_k - 1..t_k)$, nor $p^k(s_k..t_k + 1)$ is contained in phage c;*

A minimum cover *is a cover with the smallest number of intervals.*

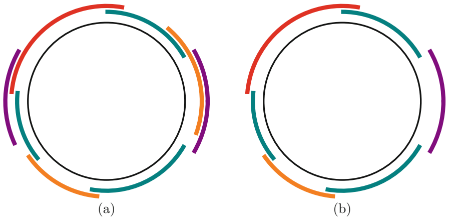


Fig. 4. (a) A cover of a circular phage, in black, by intervals of four potential parents. (b) A minimum cover extracted from cover (a).

Figure 4 gives an example of a cover and a minimum cover. The condition that no interval of a cover is contained in another implies that, in a cover, all left bound s_k are distincts and all right bounds t_k are distincts. Thus the intervals of a cover can be ordered along the circle by their distinct and increasing left bounds s_k, and we can refer without ambiguity to a pair of *consecutive intervals* of a cover $p(s_k..t_k)$ and $q(s_{k+1}..t_{k+1})$. These can be used to propose a first definition of breakpoints induced by covers:

Definition 3 (Breakpoint interval). *Given two consecutive intervals of a cover $p(s_k..j - 1)$ and $q(i..t_{k+1})$, where $i \leq j$, of a child c, then the interval $(i..j)$ of anchors is called a* breakpoint interval*.*

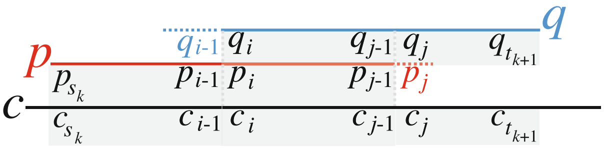

q
$q_{i\text{-}1}$ q_i $q_{j\text{-}1}$ q_j $q_{t_{k+1}}$
p
p_{s_k} $p_{i\text{-}1}$ p_i $p_{j\text{-}1}$ p_j
c
c_{s_k} $c_{i\text{-}1}$ c_i $c_{j\text{-}1}$ c_j $c_{t_{k+1}}$

Fig. 5. Breakpoint interval. All variants of modules in shaded areas are equal. A recombination of p and q with anchors $a \in (i..j)$ and b outside the interval $(s_k..t_{k+1})$ will create the interval $c(s_k..t_{k+1})$ of child c. Note that the interval $(i..j)$ corresponds either to the overlap of parents p and q, or, when $i = j$, is the single anchor shared by p and q.

Note that when $i = j$, there is a single anchor in the interval, and this corresponds to the classical notion of a breakpoint. However, breakpoint intervals can be arbitrary wide in the general case. Figure 5 illustrates the concept. Breakpoint intervals correspond to anchors that can be used to construct the union of two consecutive intervals. Indeed, a recombination of phages p and q, with anchors $a \in (i..j)$ and b outside the union of $p(s_k..j-1)$ and $q(i..t_{k+1})$, that is $b \in (t_{k+1}+1..s_k)$, will create the interval $c(s_k..t_{k+1})$ of child c. Such a recombination is said to *repair* the breakpoint interval.

Proposition 3 (Upper bound). *Let $s(c)$ be the size of a minimum cover by P for each child $c \in Q$, then there exists a recombination scenario from P to Q of length less than or equal to $R(Q) = \sum_{c \in Q}(s(c) - 1)$. Each recombination in the scenario lowers the value of $R(Q)$ by at least 1, and by at most $2|Q|$.*

Proof. A recombination can repair at most two breakpoint intervals, and there always exists a recombination that repairs one breakpoint interval of a child. In the worst case, each child will be reconstructed independently, and with recombinations that repair only one breakpoint interval, except for the last one, since a child with a minimum cover of size 2 can always be constructed in one recombination. In the best case, all children share the same two breakpoint intervals in their minimum covers, implying that $R(Q)$ may decrease by as much as 2|Q|.

From Proposition 3, we can sketch a greedy heuristics that tests all candidate recombinations and selects an optimal one. Performance, in terms of computer resources, is not a priority, unless some steps have the potential to lead to combinatorial explosions.

Greedy Heuristics

```
Input: Two populations $P$, $Q$ such that $P \to Q$, and $P \cap Q = \emptyset$
While $Q \neq \emptyset$
  1. Compute the size $s(c)$ of a minimum cover for each phage $c \in Q$, and $R(Q)$.
  2. If there exists $c$ such that $s(c) = 2$, remove $c$ from $Q$. Add the two children
     of the recombination that creates $c$ to $P$.
  3. Otherwise, find a recombination that maximizes $R(Q) - R'(Q)$, where $R'(Q)$
     is computed after simulating each possible candidate recombination. Add the
     children of this recombination to $P$.
```

Fortunately, due to applications in surveillance systems, the problem of computing minimum covers of circles has received a lot of attention. Lee and Lee [12] gave a solution in $\mathcal{O}(|S|)$ if the intervals of S are maximal elements, in terms of inclusion, and the intervals are sorted by their increasing starting points. In this case, there is at most one interval in S that begins, or ends, at any point in $(0..m-1)$. Let I be an interval in S that ends at t, its successor $succ(I)$ is the – unique – interval that contains $(t \bmod m)$, and has the largest starting point. The algorithm described in [12] finds a minimum cover by iterating the function $succ(I)$. When the iteration begins with the largest interval of S, a minimum cover is found after at most $2m$ iterations.

However, in general, minimum covers are far from unique: many pairs of parents can repair a breakpoint interval, and a child may have alternate minimum covers that do not share any breakpoint interval. Moreover, there exists the theoretical possibility that a shortest scenario must use breakpoint intervals of covers that are not minimal (see Example 1 of Annex 1).

Our first experiments with phage data appeared to yield "pretty good" results. Many recombinations were shared by two or more children, and many children were constructed using exactly $\left\lfloor \frac{s(c)+1}{2} \right\rfloor$ recombinations, where $s(c)$ is the size of a minimal cover of child c, which is a strict lower bound for a single child.

Quantifying "pretty good" without relying on minimum covers is discussed in the next section.

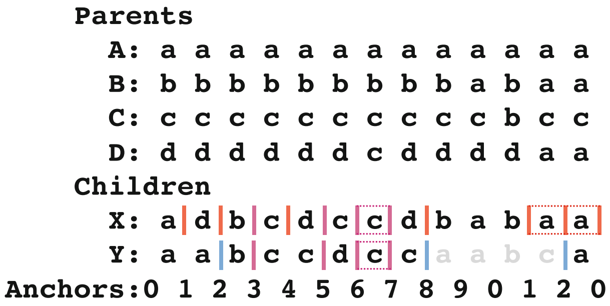
2.3 Lower Bounds

In this section, we explore under which conditions some breakpoints are *mandatory*, in the sense that they belong to any possible cover. As we will show, it is easy to construct an example with no mandatory breakpoints. Thus, a lower bound based on mandatory breakpoints has the potential to be useless in the general case, but, in datasets that come from the comparison of phages populations, they are sufficiently abundant to provide practical lower bounds.

The formal definition is based of the detection of *useful breakpoints*, that identify intervals that no single parent can cover, but that are covered by two consecutive intervals of a cover.

Definition 4. *A* useful breakpoint *is an interval $c(i-1,j)$ of a child c in Q, that is not contained in any parent of P, yet both $c(i-1,j-1)$ and $c(i,j)$ are. A useful breakpoint is thus the interval spanning a breakpoint interval together with its two flanking modules.*

A useful breakpoint can be *erased* by simply adding to the set of parents P a new parent that contains the interval $c(i-1..j)$: Example 1 of Annex 1 shows a population whose only child has 9 breakpoint intervals, but only two of them are useful. On the other hand, in our phage comparisons, almost all breakpoint intervals are useful.



```
    Parents
      A: a a a a a a a a a a a a a
      B: b b b b b b b b b a b a a
      C: c c c c c c c c c c b c c
      D: d d d d d d c d d d d a a
    Children
      X: a|d|b|c|d|c|c|d|b a b|a|a|
      Y: a a|b|c c|d|c|c a a b c|a
Anchors:0 1 2 3 4 5 6 7 8 9 0 1 2 0
```

Fig. 6. In this example child Y has 6 mandatory breakpoint. Five have a single anchor: a|b with anchor {2}, b|c with anchor {3}, c|d with anchor {5}, c|a with anchor {8}, c|a with anchor {12}; and one has two anchors d|c|c with anchors {6, 7}. One region, in gray, contains two overlapping useful breakpoint, a|a|b with anchors {9, 10}, and a|b|c with anchors {10, 11}. Breakpoints in blue are exclusive to Y, breakpoints in red are exclusive to X, and the three breakpoints in violet are *shared*.

Two sets *overlap* if their intersection is non-empty, and neither is contained into the other. When the sets of anchors of two or more useful breakpoint overlap, then there exists alternative covers of different lengths for that region. We want to eliminate this possibility:

Definition 5 (Mandatory breakpoint). *A* mandatory *breakpoint of a child c in Q is a useful breakpoint whose set of anchors does not overlap any other such set of anchors.*

In Fig. 6, all breakpoints of child X are mandatory, and we can deduce a cover from these breakpoints by constructing the sequence of parents that are exchanged at each breakpoint. In this case the only possible cover is ADBCDCDBA.

However, child Y = aabccdccaabca that has two overlapping useful breakpoint: a|a|b with anchors {9, 10}, and a|b|c with anchors {10, 11}. This yield two different covers of Y, one of length 9, ABCDCABCA, and one of length 8, ABCDCACA.

We say that two mandatory breakpoints $c(i-1, j)$ and $d(i-1, j)$ are *shared*, if they have the same set of anchors and a common pair of parents (p, q) that can repair them. Then either $c(i-1, j) = d(i-1, j)$, or $c(i-1, j)$ and $d(i-1, j)$ are a pair of twins constructed by the same recombination.

In Fig. 6, there is one pair of equal mandatory breakpoints between children X and Y at anchor {3} with parents B and C, and two pair of twins, one with anchor {5}, and one with anchors {6, 7}, both with parents C and D.

Being based on equality, the 'shared' relation is an equivalence relation on the set of mandatory breakpoints. In the example of Fig. 6, child Y has 6 mandatory breakpoints and X has 8. Three of them are shared, thus there are 11 breakpoints that must be repaired, yielding a minimum of 6 recombinations, since a recombination repairs at most two shared mandatory breakpoints (see Example 2 of Annex 1 for a recombination scenario of length 6).

In general, we have:

Proposition 4. *Let P and Q be two populations such that $P \rightarrow Q$, let M be the number of shared mandatory breakpoints of children in Q with respect to P, then the length of a shortest recombination scenario is at least $r(Q) = \left\lfloor \frac{M+1}{2} \right\rfloor$.*

3 Experiments

3.1 Dataset Construction

In order to identify core and variable genomes, we aligned the 38 genomes of the Murphy study [14] using the Alpha aligner [1]. This aligner identifies the core and variable genomes, the core becoming the anchors between the variable regions that comprise the modules. For this experiment, we used the default values of the software. Alpha was the preferred alternative to a painstaking and time consuming breakpoint analysis done by hand, where sequence similarity is queried using a tool such as BLAST. Alpha is adapted specifically to phages, and identifies major breakpoints and variants in a single automated step.

The size of the 38 genomes varied between 29 097 and 31 049 bps, with a core genome of size 7722 bps distributed across 84 anchors.

Each phage p_j from the collection receives an identifier that is a single unique letter, and a new variant for module i in phage p_j receives the identifier of p_j as its variant identifier. This yields the set of strings displayed in Fig. 2a, which is processed by the greedy algorithm. Each phage belongs to one of four factories, identified by F1, F2, F3 and F4 in the original paper. The number of phages per factory are, respectively, 13, 4, 13 and 8.

When computing a recombination scenario from a source factory to a target factory, we may encounter a variant that occurs only in the target and not in the source. We call such a variant a *missing* variant. The algorithm deals with missing variants in two ways: if stretches of missing variants are shared by two or more children in the target factory, supplementary chromosomes that contain these stretches are added to population P; if stretches of missing variants are unique to a child, the importation of each stretch counts as one recombination, and these recombinations can be done at the end of the scenario. These strategies simulate the ‘genetic reservoir’, and should be sound as long as the stretches are not too long.

3.2 Comparing Factories

Each factory was compared with the three others, resulting in 12 recombination scenarios whose length varied from 29 to 158. Table 1 presents various statistics of these scenarios, and a *gap ratio* that compares the performance of the heuristics to both the lower bound $r(Q)$, given by Proposition 4 and the upper bound $R(Q)$, given by Proposition 3. If $L \leq R(Q)$ is the length of an actual scenario, we score it with the formula:

$$\text{Gap ratio } = \frac{(R(Q) - L)}{(R(Q) - r(Q))}.$$

Gap ratios range from 0, for the worst scenario, to 1 for a scenario whose length is equal to $r(Q)$.

Since ℓ_{PQ} depends on the number of phages to reconstruct, we also give in Table 1 the average length needed to reconstruct one child.

Table 1. Comparisons of four Dutch dairies. Scenario F1 → F2 computed by the greedy algorithm has 29 recombinations, and theoretical lower bound of 24, and upper bound of 47, yielding a gap ratio of 0.78; among the 29 recombinations, 20 were used to import missing data; the size of target factory F2 is 4 phages, thus the average length of the scenario is 7.25 recombinations per phage.

Factories	Scenario Length	Lower Bound $r(Q)$	Upper Bound $R(Q)$	Gap ratio	Missing Data	Target Dairy Size	Average Length
F1 → F2	29	24	47	0.78	20	4	7.25
F1 → F3	112	88	213	0.81	77	13	8.62
F1 → F4	84	60	161	0.76	53	8	10.50
F2 → F1	158	111	358	0.81	190	13	12.15
F2 → F3	149	107	402	0.86	105	13	11.46
F2 → F4	109	80	231	0.81	79	8	13.63
F3 → F1	152	90	361	0.77	101	13	11.69
F3 → F2	53	39	90	0.73	40	4	13.25
F3 → F4	110	81	209	0.77	73	8	13.75
F4 → F1	151	109	282	0.76	100	13	11.62
F4 → F2	54	42	90	0.75	36	4	13.50
F4 → F3	151	103	370	0.82	94	13	11.62

Gap ratios range from 0.73 to 0.86, which is promising, given our very conservative lower bounds that assume that all recombinations occur between pairs of mandatory breakpoints; and given the fact that the upper bounds, based on minimum covers, cannot be lowered in the general case. Indeed, upper bounds are reached on recombination scenarios in which children do not share breakpoints, and on parents who contribute at most a single interval of contiguous modules in a child. In our dataset, specialized subpopulations were apparent in Fig. 2, implying that children of these subpopulations did not share breakpoints, and parents that contribute a single interval of contiguous modules in a child are used to model the “genetic reservoir”.

Another interesting aspect of these comparisons is that Factory F1 is obviously the best ‘constructor’, with an average of 8.79 recombinations to reconstruct all other factories, compared to 12.90 for Factory F3, for example. On the other hand, F4 is the hardest to construct, being the highest result for each of the other factories.

3.3 Shared Evolution

We tested how much shared evolution there was when creating the phages from one factory using the phages from another. To do this, we compared the scenario lengths from Table 1 to the scenario lengths obtained separately from each factory to each phage individually. The results are shown in Table 2. The table shows that, by considering the phages in the target factories simultaneously, we economize 12, 1.75, 15.7, and 7.6 recombinations per phage for target factories F1, F2, F2, and F4 respectively.

Table 2. Creating each phage individually instead of creating all phages from a factory at once. Each individual phage for a factory was created independently of the others. The sum of the lengths of the recombination scenarios creating all phages from F1, using the phages from F2, is in the first row: 61 more recombinations were required to do this as compared to creating all phages of F1 with shared recombinations. Overall, to create F1 from each of the other factories we use 13.63 recombinations more per phage.

Factories	Scenario length	Sum of individual lengths	Difference	Economy per phage
F2 → F1	158	219	61	13.23
F3 → F1	152	230	78	
F4 → F1	151	184	33	
F1 → F2	29	31	2	2.75
F3 → F2	53	55	2	
F4 → F2	54	61	7	
F1 → F3	112	131	19	15.85
F2 → F3	149	247	98	
F4 → F3	151	244	89	
F1 → F4	84	100	16	10.25
F2 → F4	109	150	41	
F3 → F4	110	135	25	

Due to the limited numbers of phages sampled from each factory, the variants from a target factory was not always present in a source factory. These missing variants played a role in our comparison, as they favored single-recombination replacement of longer stretches in the single-target comparisons. As described in Sect. 3.1, each stretch of missing variants that did not exist in any phage of a target factory were counted as a single recombination. Consider one of these stretches defined on a target factory with a single phage. Adding phages to this factory can only fragment these stretches, implying more stretches and more recombinations.

This gives single phage targets a significant advantage, in that they will have longer stretches of missing variants that each can be repaired with a single recombination. Despite this advantage enjoyed by the single phage targets, we observe large savings in all cases.

3.4 Population Structure

How distinctive are the populations within the factories? We approached this question by conducting experiments to test how random the structure is within the factories. The test statistic that we used was the sum of the scenario lengths over all pairs of factories.

The first null hypothesis was that the phages were randomly partitioned into factories of sizes 13, 4, 13, and 8. We tested this hypothesis by repeatedly (n times) partitioning the data uniformly at random into the prescribed sizes and rerunning the experiment of Sect. 3.2. This gave us the null distribution on the test statistic. A single sample T-test against this null distribution yielded a p-value lower then 10^{-6} even with $n = 20$. For larger n the p-value dropped precipitously.

The second null hypothesis we tested took into account the group structure as defined in Fig. 2b. The null hypothesis was that the phages were randomly distributed to the factories while preserving the group structure within each factory. That is, define a group structure vector [A, B, C, D, E] for a factory, containing the frequency of each group in the factory. We construct four random factories with the following frequency vectors:

$$\begin{aligned}&\mathrm{F1} = [3, 1, 3, 3, 2],\\ &\mathrm{F2} = [2, 0, 1, 0, 1],\\ &\mathrm{F3} = [5, 5, 0, 0, 0], \text{ and}\\ &\mathrm{F4} = [3, 0, 1, 1, 0].\end{aligned}$$

We run the experiment of Sect. 3.2 on n of these randomly constructed factories to obtain the null distribution on our test statistic. A single sample T-test against this null distribution for $n = 100$ gives a p-value is less than 10^{-11}.

4 Discussion and Conclusion

In this paper, we described the first combinatorial framework for the comparison of bacteriophage populations using recombinations. Our work represents a shift to a more global perspective of evolutionary analysis for bacteriophages, since the alignment-centric view breaks down in the presence of large amounts of recombination [3,18].

Our experiments show that the populations of phages sampled within the factories are not random, suggesting isolated evolution within individual factories. They also illuminate the potentially large amount of shared evolution in the recombination histories leading to the factories that we see today, supporting the genetic reservoir hypothesis.

While the application of our methods to the Dutch factories gives us insight into the relative diversity of phages in one factory with respect to another, our application is “still seriously data limited” due to the disproportionately sparse sampling of phage populations [3]. We expect the gene reservoir paradigm of variant sharing between populations to gain tremendous significance as better samplings become possible. There are signs that this time could be near, as a recent survey estimates the number of different phage species in the ocean to be more than 195,000 [8].

The problem of recombination scenario inference is a tricky one, whose computational complexity remains unknown. However, with some realistic assumptions on real data, we were able to give good solutions to comparisons arising from biological data. There are still many combinatorial problems that arise from this framework, which include increasing the lower bound—since we used a very conservative approach—and the development of better algorithms to find recombination scenarios.

Dataset

Bacteriophage genomes used in this paper identified by their one letter code, their name, and their accession number.

A	Phi17	KP793114	B	Phi13.16	KP793116
C	Phi19	KP793103	D	PhiJF1	KP793129
E	Phi43	KP793110	F	Phi4	KP793101
G	PhiG	KP793117	H	PhiA.16	KP793102
I	PhiD.18	KP793107	J	PhiL.18	KP793120
K	PhiF.17	KP793113	L	Phi5.12	KP793108
M	PhiM.16	KP793128	N	Phi109	KP793121
O	Phi93	KM091443	P	Phi129	KP793112
Q	PhiLj	KP793133	R	Phi155	KP793130
S	Phi16	KP793135	T	Phi44	KP793124
U	Phi114	KP793115	V	Phi15	KM091442
W	Phi40	KP793127	X	Phi145	KM091444
Y	Phi10.5	KP793119	Z	Phi19.3	KP793105
a	Phi19.2	KP793111	b	PhiL.6	KP793122
c	Phi4.2	KP793123	d	PhiM.5	KP793126
e	PhiF0139	KP793118	f	PhiA1127	KP793106
g	PhiC0139	KP793109	h	Phi91127	KP793125
i	PhiB1127	KP793104	j	PhiS0139	KP793134
k	PhiE1127	KP793131	l	PhiM1127	KP793132

Funding Information. AB is partially supported by Canada NSERC Grant number 05729-2014. MJM is partially supported by Canada NSERC Grant number 06487-2017. RVL acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds de recherche du Québec - Nature et technologies (FRQNT). KMS is partially supported by VIROGENESIS (EU H2020-PHC-32-2014 #634650) and the Labex NUMEV (ANR-10-LABX-20, 2017-2-46).

Annex 1

Example 1. A shortest scenario is not necessarily tied to a minimum cover.

```
Parents:              Recombination 1
  A: Xoooooooo          G: ooXX|oo|XXo
  B: oXXoooooo          H: oooo|XX|ooo
  C: oooXXoooo
  D: oooooXXoo          1: ooXX|XX|XXo
  E: oooooooXX          2: oooo|oo|ooo
  F: oXooooooX        Recombination 2
  G: ooXXooXXo          F: oX|oooooo|X
  H: ooooXXooo          1: oo|XXXXXX|o

Child:                  3: oX|XXXXXX|X
  J: XXXXXXXXX          2: oo|oooooo|o

                      Recombination 3
                        A: X|oooooooo|
                        3: o|XXXXXXXX|

                        J: X|XXXXXXXX|  *
                        2: o|oooooooo|
```

Child J has a minimum cover of size 5, namely ABCDE. Thus a shortest scenario must have at least 3 recombinations. Using the minimum cover, there is a trivial scenario of length 4, but there is an alternate one of length 3 that uses the cover AFGHGF, which is not a minimal cover.

Example 2. A recombination scenario of length 6 for the example of Fig. 6.

```
Parents
  A: aaaaaaaaaaaaa
  B: bbbbbbbbbabaa
  C: ccccccccccbcc
  D: ddddddcddddaa
Children
  X: adbcdccdbabaa
  Y: aabccdccaabca


Recombination 1
  C: ccccc|c|ccccbcc
  D: ddddd|d|cddddaa

  1: ccccc|d|ccccbcc
  2: ddddd|c|cddddaa

Recombination 2
  A: aaaaaaaa|aa|aaa
  1: cccccdcc|cc|bcc

  3: aaaaaaaa|cc|aaa
  4: cccccdcc|aa|bcc

Recombination 3
  B: bb|bbbbbbbab|aa
  3: aa|aaaaaacca|aa

  5: aa|bbbbbbbab|aa
  6: bb|aaaaaacca|aa

Recombination 4
  4: ccc|ccdccaabc|c
  5: aab|bbbbbbaba|a

  Y: aab|ccdccaabc|a *
  7: ccc|bbbbbbaba|c

Recombination 5
  2: d|ddddccd|dddaa
  5: a|abbbbbb|babaa

  8: a|ddddccd|babaa
  9: d|abbbbbb|dddaa

Recombination 6
  Y: aa|bc|cdccaabca
  8: ad|dd|dccdbabaa

  X: ad|bc|dccdbabaa *
 10: aa|dd|cdccaabca
```

References

1. Bérard, S., Chateau, A., Pompidor, N., Guertin, P., Bergeron, A., Swenson, K.M.: Aligning the unalignable: bacteriophage whole genome alignments. BMC Bioinform. **17**(1), 30 (2016). https://doi.org/10.1186/s12859-015-0869-5
2. Botstein, D.: A theory of modular evolution for bacteriophages. Ann. N. Y. Acad. Sci. **354**(1), 484–491 (1980). https://doi.org/10.1111/j.1749-6632.1980.tb27987.x
3. Brüssow, H.: Population genomics of bacteriophages. In: Polz, M.F., Rajora, O.P. (eds.) Population Genomics: Microorganisms. PG, pp. 297–334. Springer, Cham (2018). https://doi.org/10.1007/13836_2018_16
4. Cafora, M., et al.: Phage therapy against pseudomonas aeruginosa infections in a cystic fibrosis zebrafish model. Sci. Rep. **9**(1), 1527 (2019). https://doi.org/10.1038/s41598-018-37636-x
5. Castro-Nallar, E., et al.: Population genomics and phylogeography of an Australian dairy factory derived lytic bacteriophage. Genome Biol. Evol. **4**(3), 382–393 (2012). https://doi.org/10.1093/gbe/evs017
6. Chmielewska-Jeznach, M., Bardowski, J.K., Szczepankowska, A.K.: Molecular, physiological and phylogenetic traits of lactococcus 936-type phages from distinct dairy environments. Sci. Rep. **8**(1), 12540 (2018). https://doi.org/10.1038/s41598-018-30371-3

7. De Paepe, M., Hutinet, G., Son, O., Amarir-Bouhram, J., Schbath, S., Petit, M.A.: Temperate phages acquire DNA from defective prophages by relaxed homologous recombination: the role of Rad52-like recombinases. PLoS Genet. **10**(3), e1004181 (2014)
8. Gregory, A.C., et al.: Marine DNA viral macro- and microdiversity from pole to pole. Cell **177**(5), 1109–1123.e14 (2019)
9. Hatfull, G.F.: Dark matter of the biosphere: the amazing world of bacteriophage diversity. J. Virol. **89**(16), 8107–8110 (2015). https://doi.org/10.1128/JVI.01340-15
10. Kupczok, A., et al.: Rates of mutation and recombination in siphoviridae phage genome evolution over three decades. Mol. Biol. Evol. **35**(5), 1147–1159 (2018). https://doi.org/10.1093/molbev/msy027
11. Lavelle, K., et al.: A decade of streptococcus thermophilus phage evolution in an Irish dairy plant. Appl. Environ. Microbiol. **84**(10), e02855-17 (2018). https://doi.org/10.1128/AEM.02855-17
12. Lee, C., Lee, D.: On a circle-cover minimization problem. Inf. Process. Lett. **18**(2), 109–115 (1984)
13. Marcó, M.B., Moineau, S., Quiberoni, A.: Bacteriophages and dairy fermentations. Bacteriophage **2**(3), 149–158 (2012). https://doi.org/10.4161/bact.21868. pMID 23275866
14. Murphy, J., et al.: Comparative genomics and functional analysis of the 936 group of lactococcal siphoviridae phages. Sci. Rep. **6**, 21345 (2016)
15. Pedulla, M.L., et al.: Origins of highly mosaic mycobacteriophage genomes. Cell **113**(2), 171–182 (2003)
16. Skalka, A.M.: DNA replication-bacteriophage lambda. In: Arber, W., et al. (eds.) Current Topics in Microbiology and Immunology, pp. 201–237. Springer, Heidelberg (1977). https://doi.org/10.1007/978-3-642-66800-5_7
17. Swenson, K.M., Guertin, P., Deschênes, H., Bergeron, A.: Reconstructing the modular recombination history of staphylococcus aureus phages. BMC Bioinform. **14**, S17 (2013)
18. Yahara, K., Lehours, P., Vale, F.F.: Analysis of genetic recombination and the pan-genome of a highly recombinogenic bacteriophage species. Microb. Genom. **5**(8) (2019)